ENTERPRESS

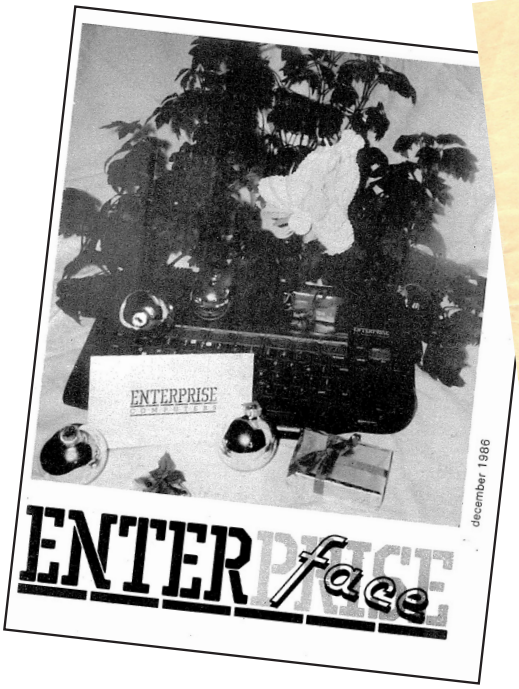
Magazin az ENTERPRISE felhasználóknak

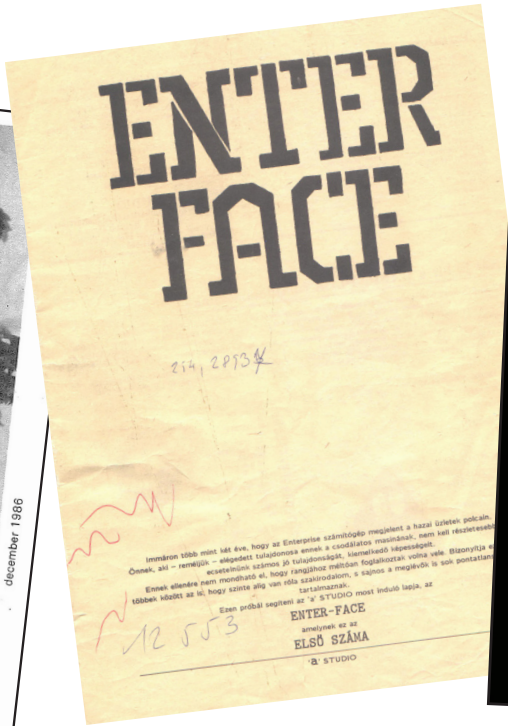
2017/5-6. szeptember-december

The Golden Baton

File kezelés Pascalban

Enterprise kiadványok a világon 1.

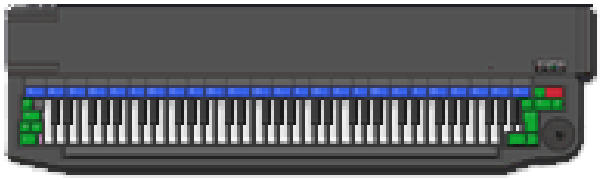




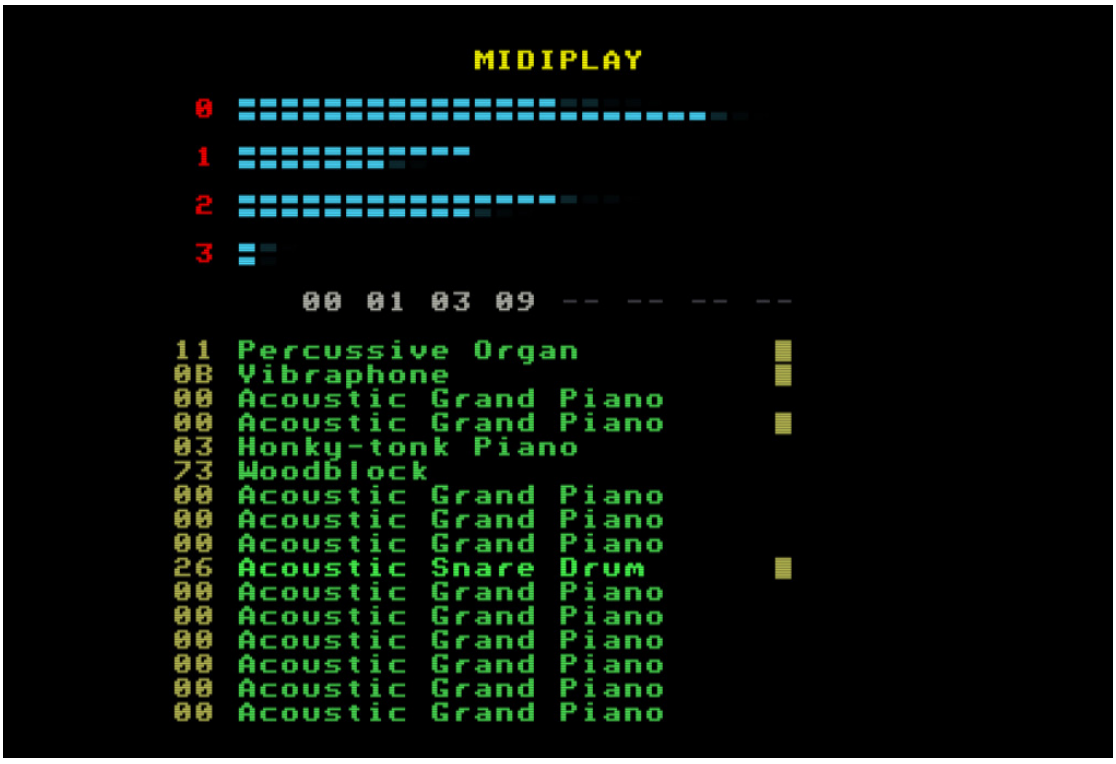
Enterprise MIDIDISP

by IstvanV

Írta: Bodnár Tamás (Szipucsu)



IstvanV elkészítette a Midiplay újabb változatát, a **MidiDisp** verziót. Ez abban különbözik az előző verziótól, hogy látható a 4 csatorna hangereje, valamint alul fel vannak sorolva az éppen használt Midi hangszerek is. Az idei évben nagy szenzáció volt a **Midiplay.** MInd emulátoron, mind Enterprise gépen használható. Emulátoron összekapcsolható akár külső Midi billentyűzettel is (ezt egyik Enterprise klubnapon ki is próbáltuk!). Bízunk benne, hogy valakinek lesz kedve Enterprise-ra egy Midi hardvert készíteni.



Z80 reset

Írta: Németh Zoltán (Zozosoft)

Sikerült egy több hónapos fejtörést okozó bugot megoldanom :-)

Az EXOS 2.4 tesztjének fejlesztése során jutottam el oda, hogy miután emulátoron remekül működött, kipróbáltam valódi gépen, és itt jött a döbbenet: reset nyomására úgy le tud fagyni, hogy nem segít a további reset nyomkodás se, csak a kikapcsolás. Néha mindjárt bekapcsoláskor ez történt.

Végül biztonsági mentésből visszaszedve az előző verziókat megtaláltam hol került be a hiba, majd összehasonlítgatva kiderült, hogy nem tervezett módon bekerült egy feltételes utasítás, anélkül, hogy előtte feltétel vizsgálat lett volna. :-) Oké, ezt én rontottam el, de miért nem jön elő emulátoron a hiba?

Rákerestem, hogy mi történik a Z80 registerekkel resetnél, és találtam egy leírást, ahol egy titokzatos Matt nevű illető információja alapján ezt írják, hogy resetnél az AF és az SP az mindig FFFFh lesz.

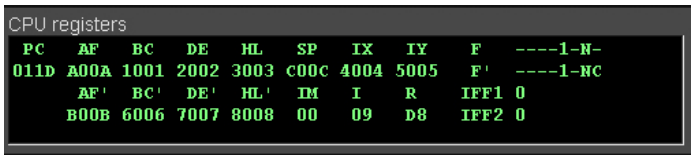
> *Matt has done some excellent research on this. He found that AF and SP are always set to FFFFh after a reset, and all other registers are undefined (different depending on how long the CPU has been powered or, different for different Z80 chips).*

Az ep128emu is így működik gondolom ezen leírás alapján (pluszban még az IX és IY regisztereket is FFFFh-ra teszi, ezt még nem találtam meg hol írták :oops:). Így érthető, hogy miért nem jött elő a hiba emulátoron.

Innen viszont egyből jön a gondolat, hogy nagyon úgy tűnik, ez a leírás hamis! Meg kéne nézni, mi is történik valójában.

Összetákoltam egy pici programot, ami alaplapi ROM-ként betéve kiírja a regiszterek értékét, majd feltölti őket értékekkel, és végtelen ciklusban várja a reset gomb megnyomását. Ekkor megtekinthető, hogy a reset milyen hatással volt a feltöltött értékekre.

Ez lesz beállítva:



Több különböző Z80 lett próbálva, de mint látható, köze nincs a leírásban állítotthoz, max néha véletlenül van olyan. A regiszterek teljesen véletlenszerűen töltődnek fel, kivéve a PC meg az I,R ami a Zilog leírás szerint is nullázva lesz. Valószínűleg valami gyártástechnikai dolog miatt, a Zilog és ST gyártmányú procik jobban szeretik az 1-es biteket, a NEC pedig a 0-akat. A legérdekesebb az NDK gyártmányú darab, ami az AAh értékeket szereti nagyon.
Reset után viszont valamennyi procin ez a helyzet:

Vagyis valamennyi register értéke megörződik, kivéve a PC meg az I,R ami nullázva lesz.

Ep128emu-n ilyen volt a helyzet:

Ha jól nézem innen:

```
void Z80::reset()
 {
   R.PC.L = 0;
   R.I = 0;
   R.IM = 0;
   R.IFF1 = 0;
   R.IFF2 = 0;
   R.RBit7 = 0;
   R.R = 0;
   R.AF.W = 0xFFFF;
   R.SP.W = 0xFFFF;
   R.IX.W = 0xFFFF;
   R.IY.W = 0xFFFF;
   R.Flags &= ~
      (Z80_EXECUTING_HALT_FLAG |
       Z80_CHECK_INTERRUPT_FLAG |
           Z80_EXECUTE_INTERRUPT_HANDLER_FLAG |
Z80_INTERRUPT_FLAG |
       Z80_NMI_FLAG | Z80_SET_PC_FLAG);
   newPCAddress = -1;
 }
```

Ezt úgy lehetett javítani, hogy ki kellett törölni a négy FFFF értékadást.

EXOS kompatíbilis memóriakezelés - IV. rész

Írta: Németh Zoltán (Zozosoft)

Povi hozzászólása:

```
                CALL VID
                JR Z,NEMHIBA
                 CP 7FH
                JP NZ,HIBA
                LD DE,200*16
                EXOS 23
                JP NZ,HIBA
                LD C,255
NEMHIBA         LD A,C
                LD (LPTS),A
NEMHIBA1        LD DE,(VIDCIM2)
                LD HL,0
                RRA
                RR H
                RRA
                RR H
                ADD HL,DE
                LD (VIDCIM2),HL
```

Itt nekem valami nem tiszta. Ha jól értelmezem a kódot, akkor LPT memória igénylés esetében, ha megosztott szegmenst kapunk, akkor a 255-ös szegmensen lesz az LPT. De miért? Miért van ott az az LD C,255?

Zozo válasza:
Az elõtte lévõ EXOS 23 hívás elrontja a C értékét, azt állítjuk vissza.

Povi hozzászólása:
Azt én értem, na de megosztott szegmens csak a 255-ös lehet?

Zozo válasza:
Jogos a kérdés, a válasz igen is meg nem is :)
Mivel 5-ös fejlécû programként indulunk, ez esetben minden csatorna le van zárva, nincs mitõl kilógjon az EXOS. Ha mondjuk rendszerbõvítõként ügyködnénk, amikor elõfordulhat megnyitott videólap, akkor lehet lentebb az EXOS határ. (Az Iview ezért modultöltéskor be is zár minden csatornát, ami nem a képfájl, hogy az EXOS „visszahúzodjon")

Ferro73 hozzászólása:
Elolvastam de nem találtam olyan lehetőséget bár már láttam valamely formáit.
Elképzelésem:

```
ld a,255
out (0b2h),a
ld hl,(0bf91h)    ;exos  alsó határ
ld a,(0bf9eh)
cp 0
jp z,nincsoszt
cp 255
jp nz,nemrencers

ld bc,hl  ;ex hl,bc ; push hl  pop bc
ld hl,0
ld de,8000h
ldir
ld a,255
out (0b0h),a
```

Mivel nem találtam utalást arra, hogy az EXOS mennyi és melyik részét használja a 255-ös szegmensből.
Gondolok a tape: disk: és egyéb átmeneti tárakra.
A fennmaradó helyen valahol csak elférne az LPT tábla és akkor felszabadul az FC szegmens.
Mivel a 0-3FFF között úgyis ROM van a ZX gépeken így nem fogja semmi zavarni a rendszer memóriát (FF, FD, FC, FE)
Ha jól emlékszem akkor a 0. ROM visszafejtése könyv a végén taglalja a rendszer szegmens terület használatát. Megvan valakinek ez a könyv? Nem találtam a teljes e-könyvet.

Zozo válasza:

> *Elolvastam de nem találtam olyan lehetőséget, bár már láttam valamely formáit.*

Az elképzelés jó, de a gyakorlati megvalósításban több hiba is van.

> *ld a,255*
> *out (0b2h),a*
> *ld hl,(0bf91h) ;exos alsó határ*
> *ld a,(0bf9eh)*

Pl. ezek máshol vannak EXOS 2.0 alatt, így nem érdemes közvetlenül vájkálni az EXOS lelkivilágában :)

”
ld bc,hl ;ex hl,bc ; push hl pop bc
ld hl,0
ld de,8000h
ldir

Hiányzik az ellenőrzés, hogy elférünk-e?

”
ld a,255
out (0b0h),a

Az EXOS-nak is meg kell mondani a változást, különben pl reset esetén az eredeti nullás lapot lapozza vissza, és kész az elszállás.

”
mivel nem találtam utalást az EXOS mennyi és melyik részét használja a 255-ös szegmensből, gondolok a tape: disk: és egyéb átmeneti tárakra.

Ez folyamatosan változik! Ezért kell lefoglalni megosztott szegmensként, majd megmondani, hogy MI mennyit használunk (felhasználói határ beállítása).

”
A fennmaradó helyen valahol csak elférne az LPT tábla és akkor felszabadul az FC szegmens. Mivel a 0-3fff között úgyis ROM van a ZX gépeken így nem fogja semmi zavarni a rendszer memóriát (FF, FD, FC, FE).

Így van, ez az amit már megcsináltam a Spectrum átiratos betöltőmben már vagy 3 éve .

Itt van kiemelve a lényeg:

```
PRGSIZE: EQU 0B00H    ; a programunk
                      ; méretéhez kell választani,
                      ; de 10H-ra kerek legyen
         LD SP,100H
         LD HL,HIBA
         LD (0BFF8H),HL
         LD A,12
         OUT (191),A
         LD BC,100H+26
         LD D,0
         EXOS 16
         LD BC,100H+28
         LD D,255
         EXOS 16
         LD BC,100H+27
         LD D,0
         EXOS 16
         HALT
         LD HL,(0BFF4H)
         SET 6,H
         LD B,4
ELPT     SRL H
         RR L
         DJNZ ELPT
         LD A,L
         LD (ELPTL),A
         LD A,H
         OR 0C0H
         LD (ELPTH),A
         LD HL,(0BFF4H)
         LD DE,STATUS
         LD BC,8
         LDIR
         CALL VID
         JP NZ,HIBA
         LD A,C
         CP 255
         JP Z,HIBA
         LD (VIDS),A
         LD DE,0
         RRA
         RR D
         RRA
         RR D
         LD (VIDCIM1),DE
         EXOS 24
         LD A,C
         LD (P2S),A
         JP NZ,HIBA
         EXOS 24
         JR Z,OKEP3
         CP 7FH
         JP NZ,HIBA
         LD A,C
         LD (LPTS),A
         OUT (0B3H),A
         CP 0FCH
         JP C,HIBA
         LD DE,PRGSIZE+200*16
         EXOS 23
         JP NZ,HIBA
         DI
         IN A,(0B0H)
         LD (P0S),A
         LD (P3S),A
         LD DE,PRGSIZE
         LD (VIDCIM2),DE
         LD HL,0
         LD DE,0C000H
         LD BC,PRGSIZE-1
         LDIR
         IN A,(0B3H)
         OUT (0B0H),A
         LD (0BFFCH),A
         LD A,(P3S)
         OUT (0B3H),A
         LD A,(LPTS)
         JR NEMHIBA1
OKEP3    LD A,C
         LD (P3S),A
         OUT (0B3H),A
         CALL VID
         JR Z,NEMHIBA
         CP 7FH
         JP NZ,HIBA
         LD DE,200+16
         EXOS 23
         JP NZ,HIBA
```

```
                LD C,255
NEMHIBA         LD A,C
                LD (LPTS),A
NEMHIBA1        CALL LPT
                LD A,(VIDS)
                OUT (0B1H),A
                LD A,(P2S)
                OUT (0B2H),A
                XOR A
                LD DE,(VIDCIM2)
                LD HL,VIDCIM2+1
                RRD
                RLCA
                RLCA
                RLCA
                RLCA
                LD (LPTL),A
                OUT (82H),A
                OR 0C0H
                RRD
                LD (LPTH),A
                OUT (83H),A
                LD (VIDCIM2),DE
```

Az LPT létrehozás, ami mint lentebb említettem változik az eredeti SpV-ben közöltekhez képest, mivel nem fix helyen van, így kezelni kell a változó videócímeket.

```
LPT:    LD DE,(VIDCIM2)
        LD HL,0
        RRA
        RR H
        RRA
        RR H
        ADD HL,DE
        LD (VIDCIM2),HL
LPT2:   LD A,(LPTS)
        OUT (0B1H),A
        LD A,192
        LD DE,(VIDCIM2)
        RES 7,D
        SET 6,D
        EXX
        LD DE,(VIDCIM1)
        LD IX,VIDCIM1
        LD HL,(VIDCIM2)
        RES 7,H
        SET 6,H
        INC HL
        INC HL
        INC HL
        INC HL
        LD BC,13
L1      EX AF,AF'
        EXX
        LD HL,LINE
        LD BC,16
        LDIR
        EXX
        LD (HL),E
        INC HL
        LD A,D
        RRA
        RRA
        RRA
        AND 3
        OR 18H
        OR (IX+1)
        LD (HL),A
        INC HL
        LD (HL),E
        INC HL
        LD (HL),D
        ADD HL,BC
        INC D
        LD A,D
        AND 7
        JR NZ,L2
        LD A,E
        ADD A,32
        LD E,A
        CCF
        SBC A,A
        AND 0F8H
        ADD A,D
        LD D,A
L2      EX AF,AF'
        DEC A
        JR NZ,L1
        EXX
        LD HL,SYNC
        LD BC,HOSSZ
        LDIR
        RET
LINE    DB 255,14H,15,2FH,0,0,0,0
TABL    DB 0,36,121,88,130,182,219,63
SYNC    DB 0F5H,2,0,0,0,0,0,0,0,0,0,0,0,0,0
STATUS  DB 247,8,11,73H,0B8H,
          0FEH,0E9H, 1,0,216, 216,0,0,0,0,0
        DB 217,12H,63,0,0,0,0,0,0,0,0,0,0,0,0
        DB 253,16,63,0,0,0,0,0,0,0,0,0,0,0,0
        DB 252,16,6,63,0,0,0,0,0,0,0,0,0,0,0
        DB 255,90H,63,32,0,0,0,0,0,0,0,0,0,0,0
        DB 252,12H,6,63,0,0,0,0,0,0,0,0,0,0,0
        DB 207,13H,63,0,0,0,0,0,0,0,0,0,0,0,0
HOSSZ   EQU $-SYNC
```

Vége

File kezelés a Pascal-ban

Írta: Povázsay Zoltán (Povi)

A HiSoft Pascal nem ismeri a file típust. A file-ok kezelésére két, nem szabványos eljárás van implementálva, amikre (bizonyos korlátok mellett) rábízhatjuk az egyszerű file-műveleteket:

A TOUT eljárással a memória tetszőleges területét menthetjük ki háttértárolóra:

tout(filename: string; address, count: integer);

Az eljárás az address memóriacímtől kezdve count bájtot ír a filename fájlba. Az eljárás automatikusan létrehozza a fájlt (ha már létező fájlt nyitunk meg, azt felülírja), majd a művelet végén lezárja azt.

A kimentett állományt a TIN eljárással olvashatjuk vissza:

tin(filename: string; address: integer);

Az eljárás az address memóriacímre betölti a filename fájlt. Sajnos a beolvasni kívánt bájtok számát nem tudjuk megadni, a fájl teljes tartalma betöltésre kerül az address címre.

Mindkét eljárás a 122-es csatornát használja, figyeljünk arra, hogy az eljárások hívása előtt a 122-es csatorna ne legyen megnyitva (magyarul: mi ne használjuk ezt a csatornát)!

Lássunk példát az eljárások használatára!

Tegyük fel, hogy van egy tömbünk, amiben a tíz legjobb játékosok neveit és pontszámait tároljuk:

```
const fn = ‚HISCORE’;

type
  TSCORE = record
    name  : packed array[1..3] of char;
    score : integer;
  end;

var
  scores : array[1..10] of TSCORE;
```

A scores tömböt az alábbi módon tudjuk kiírni a lemezre:
tout(fn, addr(scores), size(scores));
A játékunk indulása során pedig az alábbi módon tudjuk az elmentett pontszámokat betölteni:

tin(fn, addr(scores));

Mik a hátrányai a fenti két eljárásnak? A kisebbik gond az, hogy lassúak: a fájlok írása és olvasása ugyanis karakterenként történik (exos 7, illetve exos 5 használatával); ennél sokkal gyorsabb lenne a blokkműveletek használata (exos 8 és exos 6).

A nagyobbik gond a hibakezelés hiánya! A fenti példa esetében (pontszámok beolvasása lemezről), ha a lemezen még nem létezne a HISCORE nevű fájl (ami gyakorta előforduló eset, ha a játékot először indítjuk el), akkor a játékunk hibaüzenettel leállna! Jó lenne, ha az esetlegesen előforduló hibás fájlműveleteket (pl. nem létező fájl megnyitása olvasásra, írásvédett lemezre írás stb.) mi magunk tudnánk kezelni.

További hiányosságai az eljárásoknak, hogy a file-mutatót nem tudjuk pozicionálni (mindig csak a fájl elejéről tudunk olvasni, illetve csak a fájl elejére tudunk írni, nincs lehetőség létező fájl végének a hozzáfűzéséhez).

A fenti hibák kiküszöbölésére a következő számban megnézzük, hogyan készíthetnénk saját file-kezelő eljárásokat az Exos függvények használatával!

Sztringkezelő függvények megvalósítása

Írta: Kiss László (Lacika)

A Pascal nyelvben tehát egy adott karakterlánc mindig azonos számú karaktert tárol, ezt a típusa definiálásakor az indexhatár megadásakor rögzítjük. Ha mégis olyan stringet akarunk használni, amelynek hossza változik a programfutása során, akkor erre ajánlható például a következő rekord típus:

```
CONST MAXLGTH=78;
TYPE STR=RECORD
       LGTH:INTEGER;
       TEXT:ARRAY[1..MAXLGTH] OF CHAR;
END;
```

A MAXLGTH konstans a használt stringek maximális hosszát adja meg. A rekord LGTH mezőjében tároljuk, hogy pillanatnyilag a szöveg mező első hány eleme tárolja a tényleges szöveget.

Egy ilyen típusú NEV változó kényelmesen kiírható a már ismert módon:

```
WRITELN(NEV.TEXT);
```

A változó értékének beolvasásakor gondoskodnunk kell a beolvasott szövegfüzér hosszának tárolásáról:

```
PROCEDURE READSTR(VAR ST:STR);
VAR I:INTEGER;
BEGIN
  READLN;READ(ST.TEXT);
  IF (ST.TEXT[MAXLGTH]<>CHR(0)) THEN ST.LGTH:=-
MAXLGTH
  ELSE BEGIN
    I:=0;
    REPEAT
      I:=I+1
    UNTIL ST.TEXT[I]=CHR(0);
    ST.LGTH:=I-1
  END
END;
```

Ha a NEV nevű, STR típusú változóba szeretnénk beolvasni szövegfüzért, az eljárást egyszerűen így hívhatjuk meg:
READSTR(NEV);

Két azonos, STR típusú változó közötti értékadás így már egyszerű:
NEV2:=NEV;
A string-kezelő eljárások és függvényeket is elkészíthetjük, ezek közül a legegyszerűbb a változóban tárolt szövegfüzér hosszának lekérdezése:

```
FUNCTION LENGTH(ST:STR):INTEGER;
BEGIN
  LENGTH:=ST.LGTH
END;
```

Egy STR típusú változó értékét az alább eljárással törölhetjük:

```
PROCEDURE ERASESTR(VAR ST:STR);
VAR I:INTEGER;
BEGIN
  FOR I:=1 TO MAXLGTH DO
    ST.TEXT[I]:=CHR(0);
  ST.LGTH:=0
END;
```

Két STR típusú változó tartalmát összefűzhetjük a CONCAT eljárással, az eredmény az első paraméterként megadott változóban képződik.

```
PROCEDURE CONCAT(VAR ST1,ST2:STR);
VAR I:INTEGER;
BEGIN
  I:=1;
  WHILE (ST1.LGTH<MAXLGTH) AND (I<=ST2.LGTH)
DO BEGIN
    ST1.TEXT[ST1.LGTH+1]:=ST2.TEXT[I];
    ST1.LGTH:=ST1.LGTH+1;I:=I+1
  END
END;
```

A COPYSTR eljárással egy string-változó megadott részletét egy másik string-változóba másolhatjuk. Alakja:

```
COPYSTR(VÁLTOZÓ1,n1,n2,VÁLTOZÓ2)
```

A másolást az n1. karaktertől, n2. karakterig végzi, az eredmény a VÁLTOZÓ2-ben jön létre, annak korábbi tartalma felülíródik.

```
PROCEDURE COPYSTR(ST1:STR;POS1,POS2:INTE-
GER;VAR ST2:STR);
VAR I:INTEGER;
BEGIN
  ERASESTR(ST2);
  IF POS1<1 THEN POS1:=1;
  IF POS2>ST1.LGTH THEN POS2:=ST1.LGTH;
  FOR I:=POS1 TO POS2 DO
    ST2.TEXT[I]:=ST1.TEXT[I];
  ST2.LGTH:=POS2-POS1+1;
  IF ST2.LGTH<0 THEN ST2.LGTH:=0
END;
```

A DELSTR eljárással egy string-változó tartalmából törölhetjük a megadott részletet. Alakja: DELSTR(VÁLTOZÓ,n1,n2). A törlést az n1 karaktertől n2 karakterig végzi. A COPYSTR és DELSTR eljárás „bolondbiztos", azaz ellenőrzi a megadott paramétereket.

A DELSTR eljárás felhasználásával könnyen elkészíthetjük az LTRIM eljárást is, amely egy szövegfüzér elején álló szóközöket törli.

Az RTRIM eljárás - mely a szövegfüzér végén lévő szóközöket törli - hasonlóképen megoldható:
Az UCASE eljárás egy string-változót nagybetűsít (a kisbetűket nagybetűkre cseréli, minden más karaktert változatlanul hagy):

Az LCASE eljárás egy string-változót kisbetűsít:

```
PROCEDURE LCASE(VAR ST:STR);
VAR I:INTEGER;
BEGIN
  FOR I:=1 TO ST.LGTH DO
    IF ST.TEXT[I] IN [,A'..'Z'] THEN
      ST.TEXT[I]:=CHR(ORD(ST.TEXT[I])+32)
END;
```

A POS föggvény az ST2 változóban tárolt szövegrészlet első előfordulásának pozícióját keresi meg az ST1 változóban tárolt szövegfüzérben. Ha a keresett szövegrészlet nem található, a függvény visszaadott értéke nulla.

```
FUNCTION POS(ST1,ST2:STR):INTEGER;
VAR I,J:INTEGER;
BEGIN
  I:=1;J:=1;
  REPEAT
    IF ST1.TEXT[I]=ST2.TEXT[J] THEN J:=J+1 ELSE J:=1;
    I:=I+1;
  UNTIL (I>ST1.LGTH) OR (J>ST2.LGTH);
  IF J=ST2.LGTH+1 THEN POS:=I-J+1 ELSE POS:=0
END;
```

Tetszőleges karaktersorozatot tartalmazó, de nem nulla hosszúságú (üres) karakterfüzért INTEGER típusú számmá alakíthatunk pl. a következő függvénnyel. Az egyszerűség miatt a legnagyobb konvertálható szám 32759.

```
FUNCTION VAL(ST:STR):INTEGER;
VAR I,SZ:INTEGER;
    N:BOOLEAN;
BEGIN
  I:=0;SZ:=0;N:=FALSE;
  REPEAT
    I:=I+1;
    IF ST.TEXT[I]='-' THEN N:=TRUE;
  UNTIL (ST.TEXT[I] IN [,0'..'9']) OR (I=ST.LGTH);
  IF ST.TEXT[I] IN [,0'..'9'] THEN BEGIN
    SZ:=ORD(ST.TEXT[I])-48;
    FOR I:=I+1 TO ST.LGTH DO
      IF (ST.TEXT[I] IN [,0'..'9']) AND (SZ<3276) THEN
        SZ:=SZ*10+(ORD(ST.TEXT[I])-48)
  END;
  IF N THEN VAL:=-SZ ELSE VAL:=SZ
END;
```

Rendszerszegmens memóriatérképe

- nem EXOS kompatíbilis programozáshoz

EXOS 2.0 Address	EXOS 2.1 Address	Name
(BF97h)...(BF99h)-1	(BF91h)...(BF93h)-1	Channel area
(BF99h)...(BF9Bh)-1	(BF93h)...(BF95h)-1	Device descriptors
(BF9Bh)...(BF9Eh)-1	(BF95h)...(BF97h)-1	RAM areas allocated to the ROM extensions
(BF9Eh)...(BFA0h)	(BF97h)...(BF9Ch)	Extension ROM list
(BFA0h)+1...(BF9Eh)...ABD5h	(BF9Ch)+1...(BF9Ah)...ABD0h	RAM segment list
ABD6h...	ABD1h...	EXOS work area
...B21Bh	...B216h	Z80 Stack under EXOS operations
B21Ch...B234h	B217h...B22Fh	EXOS paging routines
B235h...B256h	B230h...B251h	„Written by: Mrl BT NMV GNH CGE AEL"
		Devices work area
B680h...BAFFh	B480h...B8FFh	Character Font (128 characters)
(BFF4h)BB00h...BD1Fh	(BFF4h) B900h...BB1Fh	Line Parameter Table
		EXOS work area
BEBCh...BEE3h	BEB8h...BEDFh	Status Line
		EXOS work area
BF1Dh...	BF18h...	Default Device
		EXOS work area
BF76h	BF72h	Time: Second (BCD)
BF77h	BF73h	Time: Minute (BCD)
BF78h	BF74h	Time: Hour (BCD)
BF79h	BF75h	Date: Day (BCD)
BF7Ah	BF76h	Date: Month (BCD)
BF7Bh	BF77h	Date: Year (BCD), started at 1980
		EXOS work area
BF82h	BF7Eh	ROM CRC for secret protection routine
		EXOS work area
BF97/8h	BF93/4h	Bottom of the Device descriptors
BF99/Ah	BF95/6h	Bottom of the RAM area allocated to the ROM extensions
BF9B/Ch	BF97/8h	End Pointer of the ROM list
BF9Dh	BF99h	
BF9E/Fh	BF9A/Bh	Pointer of the RAM list
BFA0/1h	BF9C/Dh	Start Pointer of the ROM list
BFA2h	BF9Eh	Segment number of the SHARED segment (0 if none)
BFA3h	BF9Fh	Number of free segments
BFA4h	BFA0h	Number of USER allocated segments
BFA5h	BFA1h	Number of DEVICE allocated segments
BFA6h	BFA2h	Number of SYSTEM segments
BFA7h	BFA3h	Number of working segments
BFA8h	BFA4h	Number of defective segments
BFBE/F/C0h	BFBA/B/Ch	First pointer of the Channel chain
BFC1/2/3h	BFBD/E/Fh	First pointer of the Device chain
BFC4/5/6h	BFC0/1/2h	First pointer of the System Extension chain
BFC9h	BFC5h	IRQ_ENABLE_STATE
BFCAh	BFC6h	

BFCBh	BFC7h	CODE_SOFT_IRQ
BFCCh	BFC8h	DEF_TYPE
BFCDh	BFC9h	DEF_CHAN
BFCEh	BFCAh	TIMER
BFCFh	BFCBh	LOCK_KEY
BFD0h	BFCCh	CLICK_KEY
BFD1h	BFCDh	STOP_IRQ
BFD2h	BFCEh	KEY_IRQ
BFD3h	BFCFh	RATE_KEY
BFD4h	BFD0h	DELAY_KEY
BFD5h	BFD1h	TAPE_SND
BFD6h	BFD2h	WAIT_SND
BFD7h	BFD3h	MUTE_SND
BFD8h	BFD4h	BUF_SND
BFD9h	BFD5h	BAUD_SER
BFDAh	BFD6h	FORM_SER
BFDBh	BFD7h	ADDR_NET
BFDCh	BFD8h	NET_IRQ
BFDDh	BFD9h	CHAN_NET
BFDEh	BFDAh	MACH_NET
BFDFh	BFDBh	MODE_VID
BFE0h	BFDCh	COLR_VID
BFE1h	BFDDh	X_SIZ_VID
BFE2h	BFDEh	Y_SIZ_VID
BFE3h	BFDFh	ST_FLAG
BFE4h	BFE0h	BORD_VID
BFE5h	BFE1h	BIAS_VID
BFE6h	BFE2h	VID_EDIT
BFE7h	BFE3h	KEY_EDIT
BFE8h	BFE4h	BUF_EDIT
BFE9h	BFE5h	FLG_EDIT
BFEAh	BFE6h	SP_TAPE
BFEBh	BFE7h	PROTECT
BFECh	BFE8h	LV_TAPE
BFEDh	BFE9h	REM1
BFEEh	BFEAh	REM2
Not exist	BFEBh	SPRITE
Not exist	BFECh	RANDOM_IRQ
Not exist	BFED/Eh	USER_ISR
BFEFh		CRDISP_FLAG
BFF0/1h		SECOND_COUNTER
BFF2h		FLAG_SOFT_IRQ
BFF3h		PORTB5
BFF4/5h		LP_POINTER
BFF6/7h		ST_POINTER
BFF8/9h		RST_ADDR
BFFA/Bh		STACK_LIMIT
BFFCh		USR_P0
BFFDh		USR_P1
BFFEh		USR_P2
BFFFh		USR_P3

The Golden Baton

1983 - Digital Fantasia - kaland, szöveges

Írta: Kiss László (Lacika)

A 2017/2-3. számban megjelent „játéktörténeti összefoglalóban" csak egy rövid félmondatban emlékeztünk meg Brian Howarth munkásságáról, azon belül a Mysterious Adventures sorozatról. A most induló cikksorozatban a sorozat játékairól „emlékezünk meg", egyrészt azért mert a szöveges kalandjátékok történetében jelentős mérföldkőnek számítanak, ennek ellenére magyar nyelvű ismertető még nem jelent meg róluk sehol, másrészt azon el nem hanyagolható oknál fogva, hogy Geco software-es emulátorával tökéletesen futnak ezen játékok (beleértve a játékállás mentését / töltését)!

Brian Howarth 1981-ben hagyta ott telekommunikációs mérnöki karrierjét, hogy a Dungeons & Dragons szerepjáték, az Adventure és a korabeli számítógépes újságokban megjelent BASIC nyelvű begépelhető, egyszerű játékoktól kedvet kapva szöveges kalandjátékok készítésébe kezdjen. A korabeli BASIC nyelvű játékok azonban nem igazán nyerték el a tetszését, ezért - grafika nélküli, kezdetleges - játékait gépikódban írta TRS-80 mikrogépre. Egy, az Infocom „virtuális számítógépéhez" hasonló megoldás, amit ő interpreter-nek nevezett lehetővé tette, hogy alig két év alatt 11 kalandjátékot készítsen: a játék egy adat file-ban volt, az azt futtató interpreter nem változott.

A fejlettebb grafikus képességekkel rendelkező otthoni számítógépek megjelenése után játékai a Mysterious Adventures című sorozatban is megjelentek ezen gépekre (Atari, Acorn Electron, BBC Micro, Commodore 64, Commodore Plus/4). A Spectrum verziók 1983-ban készültek. Akkor ezek a játékok még egészen különlegesnek számítottak: teljes egészében gépikódban készültek - ez akkor még egyáltalán nem volt általános - és a Spectrum és C64 verziók valamennyi helyszínről grafikát is adtak (a korabeli kalandjátékok többsége még egyáltalán nem tartalmazott grafikát...). Ennek azonban ára volt: mivel elég sok helyszín tartogatott a játék, a grafikát nem lehetett a programban tárolni, a „sematikus rajzokat" vonalakból rajzolta ki a program, majd a zárt alakzatokat egy - meglehetősen lassú - kitöltő rutin színezte ki. Részben a képek nagy helyigénye miatt még így is jelentősen redukálni kellett a szókincset és a helyszínek szöveges leírása is tőmondatokra korlátozódik. A játékok csak az ige + főnév szóösszetételt értették meg - ez akkoriban teljesen természetes volt. Az azonban már frusztráló lehet, hogy a rendkívül alacsony szókincs miatt szinte egyetlen szinonimát sem ismer a program, s majd' minden igére, tárgyra egyetlen szó létezik csak. Ennek következében több idő megy el a program által ismert szavak keresgélésére, mint a tényleges játékra. Ez - bár az angolul kevésbé értők akár előnynek is felfoghatják - sajnos erősen rontja az amúgy érdekes történetet és ötletes rejtvényeket tartalmazó játékok játszhatóságát. Érdekes, bár kevésbé hasznos funkció, hogy a sorozat összes tagja támogatja a Currah MicroSpeech Speaker rendszert, amivel a program „felolvassa" a történteket.

A Mysterious Adventures sorozatban megjelent 11 játék azonban jó időben jelent meg, így minden hibája ellenére klasszikussá vált:

A két részben megjelenő Arrow of Death ugyanabban a világban játszódik, mint a Golden Baton, ebben a gonosz Xerdon-t kell egy mágikus nyílvesszővel megölnünk.

Az Escape from Pulsar 7 egy sci-fi kalandjáték, melyet erősen inspirált az Alien film.

A Feasibility Experiment az Adventureland-hez hasonló kincsgyűjtögetős játék.

A Time Machine című részben egy tudós, Dr. Potter megmentésére indulunk térben és időben.

A Circus-ban egy kísértetjárta cirkuszból kell elmenekülnünk.

A Wizard Akyrz című részben a címben szereplő varázslóval gyűlik meg a bajunk, miután elrabolja uralkodónk leányát és ellopja a kincstár értékeit.

A Perseus and Andromeda a görög mitológiába vezet be - Perszeusz és Andromeda történetével.

A Ten Little Indians című részben egy nemrég elhunyt műgyűjtő birtokán kell felkutatnunk egy rendkívül értékes műtárgyat.
A Waxworks szürreális történetében feléledt viaszfigurák kergetnek minket.
A sorozat első darabja a Golden Baton-ban a címben szereplő arany varázspálca körül bonyolódnak az események. Ez a felbecsülhetetlen értékű mágikus tárgy tartja fent a jó és a rossz közötti egyensúlyt a Ferrenuil királyságban. Egy napon azonban ellopták a király palotájából. A jó és a rossz egyensúlyának felborulása aszályban és pestisjárványban nyilvánul meg. Feladatunk az arany varázspálca felkutatása és megszerzése.
Akik egyedül szeretnének próbálkozni a játékkal, azoknak először összefoglaljuk a programok kezelését.
Mozogni az alábbi parancsokkal tudunk:
GO NORTH, N
GO WEST, W
GO SOUTH, S
GO EAST, E
GO DOWN, D
GO UP, U
A GO parancs után helyszín neve is állhat (amire hivatkozik a leírásban a program).
A játék kezelését segítő parancsok:
SAVE - játékállás kimentése,
QUIT (rövidítve: Q) - Kilépés a játékból. A program megerősítést vár, majd megkérdezi, hogy akarunk-e új játékot kezdeni. Minden új játék indításakor - így a program betöltése után is - a program megkérdezi, kívánunk-e játékállást betölteni. Korábban elmentett játékállás betöltésére nincs is más mód (nincs LOAD parancs).
INVENTORY (rövidítve: I): a játékosnál lévő tárgyak listája.
SCORE: A „szokásjog" alapján maradt csak a programban, mindössze annyit ír ki, hogy ez nem futball mérkőzés...
HELP: általában triviális dolgokat puffogtat: próbáljuk felfedezni a környéket, vizsgáljunk meg mindent. Néha azonban tényleg segít!
Az ENTER megnyomásával ki-be kapcsolhatjuk a képek megjelenítését. Kikapcsolt grafikával rövid szöveges leírást kapunk a helyszínről, illetve az ott lévő tárgyakról. Mivel ezt a leírást eltakarja a kép, folyamatosan váltogatni kellene a grafika és a leírás között. Azonban az amúgy is ronda (legnagyobb jóindulattal is csak „sematikus") grafika kirajzolása eléggé lassítja a játékot, érdemes inkább kikapcsolt grafikával játszani.
A program további szókészlete:

AKRYZ	FEED	LIZARD	RUNES
ARCH	FILL	LOOK	SAIL
AROUND	GET	MATCHES	SALT
ATTACK	GIVE	MIRROR	SAY
BARREL	HAMMER	OPEN	SEARCH
BATON	HELMET	PARCHMENT	SLUGS
BLOW	HIT	PATH	STABLES
BRIARS	HOLD	PILE	STAFF
BURN	HOLE	POLISH	STRAW
CABIN	HOLLOW	PORTCULLIS	STREAM
CAVE	HORN	POUR	SWIM
CHOP	JUMP	PUT	SWORD
CLIMB	KEY	QUARTZ	TAKE
CLOAK	KILL	RAFT	THROW
CRAB	KNIFE	READ	TREE
DOOR	LAKE	REMOVE	UNLIGHT
DROP	LAMP	RING	WAVE
EAT	LEAVES	ROPE	WEAR
EXAMINE	LIGHT	RUB	WOLF

A szavak rövidíthetők, elég az első négy karakterüket begépelni. Megjegyzendő, hogy egyszerre öt tárgyat bírunk el.
Egy lehetséges megoldás:
Egy kísértetjárta sűrű erdőben kezdjük kalandunkat (1. kép). Egy viseletes köpönyeg hever itt, és egy rothadó levélkupac. A köpönyeg még jól jöhet, a levélkupac nem tűnik túl érdekesnek, de abból baj nem lehet, ha megvizsgáljuk (GET CLOAK, WEAR CLOAK, EXAMINE LEAVES). Nem is tévedünk, egy csillogó kardot találunk a kupac alatt (GET SWORD, vagy TAKE SWORD)! Délre (S), ahol sűrű tövisbokor zárja el az utat, zsákutcába jutottunk. Ha megnézzük közelebbről (LOOK BRIARS), a program megerősíti, hogy valóban szúrós. Van viszont egy kardunk, azzal kicsit megritkíthatjuk (CHOP BRIARS), találunk is egy kötelet a tüskék között (GET ROPE).
Induljunk el északra (N, N, N). Egy tisztásra jutottunk ahol egy farkas néz velünk „farkasszemet". Látunk ugyan egy ösvényt, de a farkas nem enged... Megetetni (FEED WOLF) nem tudjuk, marad a kard (ATTACK WOLF vagy KILL WOLF). A kardra már nem lesz szükségünk (DROP SWORD). Egyelőre még ne az ösvényen, hanem még visszafelé menjünk (S, W). Egy idős, hatalmas tölgyfa alatt állunk. Bár a fát megvizsgálva semmit nem segít a program, ha lasszó módjára feldobhatjuk rá a kötelet, megakad az egyik ágában és felmászhatunk rajta (THROW ROPE, CLIMB TREE). A fa törzsén egy odút találunk, ami egy gyűrűt rejt (LOOK HOLLOW, GET RING). Ha megvizsgáljuk a gyűrűt (EXAMINE RING), megtudhatjuk, hogy rúnajelek vannak rávésve. Abból baj nem lehet, ha felhúzzuk az ujjunkra (WEAR RING). Másszunk vissza a földre, majd bravúros mozdulattal felvehetjük a kötelet (D, GET ROPE). Most már visszamehetünk az ösvényhez (E, N, GO PATH). Egy erdei úton haladunk, lábunk előtt egy bot hever (GET STAFF). Ha megnézzük közelebbről (EXAMINE STAFF), ezen is rúnajelek vannak, olyanok, mint a gyűrűn.
Az ösvény északra vezet (N, N, N), míg egy vár várárok nem állja utunkat. A várárokhoz persze vár is tartozik, be kellene jutni. A THROW ROPE nem működik, és a GO MOAT sem (ez utóbbira csak egy „Sorry" feliratot kapunk), csak a (SWIM) segít. A kapurostély közelében lubickolunk, a „balga" kalandor mindent el fog követni, hogy kinyissa a rácsot, ami persze nem fog sikerülni, és egy óra próbálkozás után otthagyja a játékot, miután az INVENTORY parancs kiadása után megállapítja, hogy a gyufája is elázott... Nem túl logikus módon a vízben kapálózva már fel tudjuk dobni a kötelet a várfalra, így már felmászhatunk a vár oromzatra (2. kép) (THROW ROPE, CLIMB ROPE). Délre a kapun keresztül kisétálhatunk a várból (ki engednek, csak be nem), de inkább a másik opciót válasszuk: le a várudvarba (D). Egy félelmetes fekete páncélos lovag őrködik itt, aki csak akkor enged át a boltíves átjárón, ha a köpeny rajtunk van (GO ARCH).
Örömünk nem tart sokáig, egy masszív, zárt ajtó állja is-

mét utunkat. De legalább nem csak visszafelé délre mehetünk, hanem keletre és nyugatra is. Keletre (E) egy istállóba jutunk, ahol ugyan ló nincs, viszont egy elefántcsont vadászkürt és egy rozsdás sisak igen. Ha eddig megpróbáltuk elolvasni a boton lévő rúnajeleket, annyit tudtunk volna csak meg, hogy olvashatatlanok. Ha viszont viseljük a sisakot (GET HELMET, WEAR HELMET), már el tudjuk olvasni (READ STAFF, vagy READ RUNES), a mágikus szót: „AKRYZ". Bár sok (semennyi) logika nincs a történtekben, a „balga" játékos ezután magával cipeli a hasznosnak tűnő sisakot, de hiába, később semmire nem tudjuk használni... (REMOVE HELMET, DROP HELMET). Csak visszafelé mehetünk (W). Kreatívabbak talán már próbálkoztak a varázsgyűrűnek tűnő gyűrű „birizgálásával", esetleg meg is dörzsölték. Akik eddig nem tették, most próbálják meg (RUB RING, vagy POLISH RING). A gyűrűben egy dzsinn „lakik", ami valami hasznosnak tűnő dologgal ajándékoz meg. Most a vizes gyufát szárítja meg. Ez mindenképen hasznos, hisz nem nehéz kitalálni, hogy a közeljövőben használni fogjuk. Nézzük, mit tud még a dzsinn (RUB RING): egy kulcsot ad (ha még nem nedves a gyufa, a kulcsot adja előbb). Sajnos ezzel ki is fogyott a dzsinn repertoárja, harmadszorra nem történik semmi (REMOVE RING, DROP RING). A kulcsot viszont épp ennél az ajtónál fogjuk használni (GET KEY, UNLOCK DOOR, DROP KEY, OPEN DOOR). A várban a köpenyre sincs szükségünk (REMOVE CLOAK, DROP CLOAK).

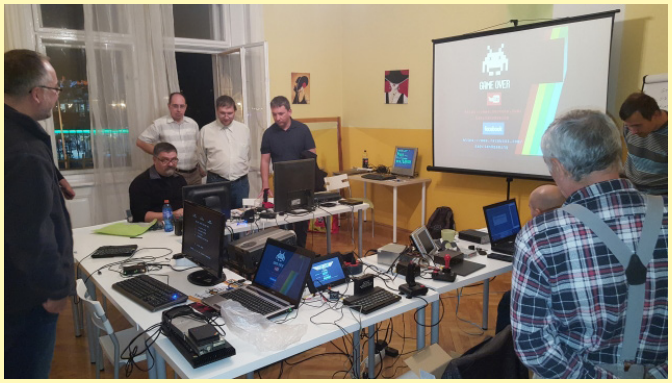
A „balga" kalandor lelkesen lép be az ajtón, ahol egy bizonyos Gorgon nevű varázsló „merev" tekintetével menten kővé változtatja őket. A megfontolt kalandor viszont előbb a (HELP) parancscsal azzal a kegyben részesül, hogy a program kivételesen segít: „nem ártana egy tükör". Ezért inkább megnézi mi van nyugatra (W). Egy fészert találunk, benne szétszórt szénát és egy olajlámpát (GET LAMP). Nézzük, mit rejt a széna (LOOK STRAW): egy lyukat! Odalenn természetesen sötét van, gyújtsuk meg a lámpát (LIGHT LAMP) - ami természetesen csak száraz gyufával lehetséges - majd mehetünk le (GO HOLE). Egy kamrába jutottunk, ahonnan egy ajtó nyílik (GO DOOR) egy hangulatos kínzókamrába (TORTURE-CHAMBER). Északra (N) a szolgák szobájába jutunk (3. kép) (ki volt ebben a várban a lakberendező???), ahol egy tükör szúr szemet (GET MIRROR).

A kínzókamrától délre (S, S), a varázsló dolgozószobáját találjuk, itt egy izzó kvarckristályon akad meg a szemünk. Az „izzót" szó szerint kell érteni, ha megpróbáljuk felvenni, nem tudjuk, mert túl forró (LOOK QUARTZ). A (HELP) azt javasolja, próbálkozzunk mágiával és gesztikulálással. Most a másik varázslatosnak tűnő szerszámunkat, a botot kell bevetni, gesztikulálásra meg sok szó nincs a szókészletben... (WAVE STAFF). A program válasza: „Valami történik!" Kösz... A kvarckristály még mindig forró így varázstudományunk utolsó darabját, a varázsszót is be kell vetnünk (SAY AKYRZ). A fehér lángnyelvek között lebegő kvarckristály az asztalra lebeg (DROP STAFF, GET QUARTZ). Keletre (E) egy sötét szobában egy groteszk gyíkemberrel találkozunk. Ha megfogadjuk a program korábbi tanácsát, miszerint vizsgáljunk meg mindent (EXAM LIZARD) megállapíthatjuk, hogy gyíkember nem túl barátságos. Legalábbis abból, hogy szó nélkül megöl bármilyen ténykedésünkre, erre lehet következtetni. A (HELP) parancs viszont ismét segít: a kvarckristály használata hasznos lehet itt (WAVE QUARTZ): egy káprázatos fénysugár elpusztítja gyíkembert! Ha átkutatjuk a tetemet (EXAMINE LIZARD), egy ékkövekkel díszített kést találunk nála. Ne habozzunk kifosztani egy hullát (DROP QUARTZ, GET KNIFE)! A kínzókamrában (W, N) már bizonyára felkeltette figyelmünket egy nagy kalapács (GET HAMMER).

A tükörrel már visszamehetünk a nagy ajtóhoz (W, U, UNLIGHT LAMP, E). Hogyan védekezzünk, a varázsló „dermesztő" tekintetétől? Azt tudjuk, hogy a tükröt kell használni, tartsuk hát magunk elé és úgy menjünk be (HOLD MIRROR, GO DOOR). A szoba tele van egészalakos szobrokkal... Ráadásul Gorgon saját tekintetével a tükörből saját magát is kővé változtatta. A veszély tehát elmúlt (DROP MIRROR), és elolvashatjuk az itt talált pergamentet (GET PARCHMENT, vagy GET PARC, READ PARCHMENT). Csak töredékek olvashatóak: „Hajózz a tavon... fújd meg a kürtöt... dob...» Kürtöt már láttunk, ezek szerint célszerű lesz magunkhoz venni (DROP PARCHMENT, S, E, GET HORN). Ezzel a vár felderítését (kifosztását) befejeztük, távozhatunk (W, S, U, S, S).

Menjünk vissza dél felé az ösvényen, egészen az útkereszteződésig, ahol már többször megfordultunk (4. kép) (S, S, S, S, S). Innen nyugatra (W) van a tölgyfa, amire felmásztunk, északra (N) tisztás egy kis házikóval (5. kép). Természetesen bemegyünk (GO CABIN). Találunk itt egy olajjal átáztatott rongyot, ha esetleg a várban annyit mászkáltunk volna, hogy kialudt a lámpa, itt feltölthetjük (FILL LAMP). Érdekesebb a padlón lévő lyuk, amelyben egy lépcső vezet lefelé és a szoba közepén álló hordó (EXAMINE BARREL), mint kiderül, só van benne. A só biztos jó lesz valamire, odalent is bizonyosan sötét van, viszont tele a kezünk (LIGHT LAMP, DROP MATCH, GET SALT, D). A délre vezető járat felé véve az irányt (S), egy belakatolt ajtó állja utunkat. A (HELP) parancs azt tanácsolja ne szöszmötöljük, zúzzuk össze a lakatot, ez a kalapáccsal nem is fog gondot okozni (SMASH PADLOCK, - vagy HIT PADLOCK -, DROP HAMMER, GO DOOR). Egy raktárhelyiségbe jutottunk, ahol egy kis tutajt találunk. Olvastunk valami tóról, bizonyára jól fog jönni egy úszó alkalmatosság (GET RAFT). Csak visszafelé mehetünk, északra az alagútban (N, N), ahol nagy zöld meztelencsigák mászkálnak. Fuj! Só és meztelencsigák? Talán nyilvánvaló az összefüggés (GIVE SALT, vagy DROP SALT). Vegyük is fel a döglött csigákat (GET SLUGS). Tovább északra (N, N) egy óriási rák állja el az utunkat, nem enged át, pedig mögötte már a tóra nyíló barlangi kijárat van (6. kép). A (HELP) szerint talán éhes, de „lekenyerezni" nem lehet... Talán a sós csigát szereti! (Ha a csigákat nem vettük fel, a (HELP) el is árulja, hogy a rákok szeretik a sózott meztelencsigát. Csiga-chips...) (FEED CRAB, vagy GIVE SLUGS, de a DROP SLUGS is jó). A rák jóllakottan félrevonul.

A parádés végkifejletben a pergamenten olvasható teendőink vannak: hajózzunk be (GO LAKE vagy SAIL LAKE), majd fújjuk meg a kürtöt (BLOW HORN). Váratlan fordulat következik: egy hatalmas kéz jelenik meg, amely az ominózus varázspálcát szorongatja. Nem tudjuk kikapni a kézből a pálcát, mert nagyon izgága... Ha hozzávágjuk a kést, akkor viszont lenyugszik (THROW KNIFE, GET BATON). A varázspálca megszerzése után a program gratulál a feladat teljesítéséhez.



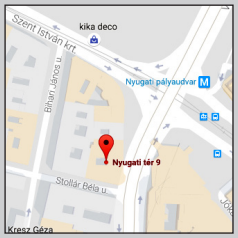
Enterprise Klub

2017. október 28.

ENTERPRISE KLUB

Egy évben 6 alkalommal

Helyszín:
Skála terem
Budapest (V. ker.)
Nyugati tér 9.
14 órától 19 óráig



Információ: www.enterpriseklub.hu

Ha te is szeretnél Az ENTERPRESS Magazin szerkesztője lenni, küldj cikket, játékleírást, játékismertetőt, vagy bármit amely az Enterprise számítógéppel kapcsolatos!

A cikkeket erre az e-mail címre küldheted:

info@enterpress.news.hu

ENTERPRISE FOREVER

https://enterpriseforever.com

ENTERPRESS Magazin - 2017/5-6. szeptember–december
Főszerkesztő: Matusa István
Szerkesztőségi főmunkatárs: Németh Zoltán (Zozosoft)
A csapat: geco, Povi, Kiss László, SzörG, szipucsu, lgb
Design, nyomdai előkészítés: Matusa István
Weboldal: http://enterpress.news.hu
E-mail: info@enterpress.news.hu
A lap időszakosan - korlátozott példányszámban - nyomtatott formátumban és elektronikus formában is megjelenik.
ENTERPRESS e-magazinok:
http://enterpress.news.hu/index.php/magazin